NAME
ra – read **argus(8)** data.

COPYRIGHT

SYNOPSIS
ra [raoptions] [- filter-expression]

DESCRIPTION
Ra reads **argus(8)** data from either *stdin*, an *argus-file*, or from a remote data source, which can either be an *argus-server*, or a netflow data server, filters the records it encounters based on an optional *filter-expression* and either prints the contents of the **argus(5)** records that it encounters to **stdout** or appends them into an **argus(5)** datafile.

OPTIONS

−A   Print aggregate statistics for the input stream on termination.

−b   Dump the compiled transaction-matching code to standard output and stop. This is useful for debugging filter expressions.

−c <char>
Specify a delimiter character for output columns (default is ' ').

−C <[host]:portnum>
Specify a source of Netflow data. The optional host is the local interface address where Netflow Cisco records are going to be read. If absent, then it is implied that the interface address is AF_ANY.

−D <level>
Print debug information corresponding to **<level>** to stderr, if program compiled to support debug printing. As the level increases, so does the amount of debug information **ra(1)** will print. Values range from 1-8.

−e <regex>
Match regular expression in flow user data fields. Prepend the regex with either "s:" or "d:" to limit the match to either the source or destination user data fields. Examples include:
   "ˆSSH-"   - Look for ssh connections on any port.
   "s:ˆGET"   - Look for HTTP GET requests in the source buffer.
   "d:ˆHTTP.*Unauth" - Find unauthorized http response.

−E <file>
When using a filter expression at the end of the command, this option will cause **ra(1)** to append the records that are rejected by the filter into **<file>**

−F <conffile>
Use **<conffile>** as a source of configuration information. The format of this file is identical to **rarc(5)**. The data read from **<conffile>** overrides any prior configuration information.

−h   Print an explanation of all the arguments.

−M <mode [mode ...]>
Provide addition mode operators. These are generally specific to the individual ra* program, or a specific function. Available modes for ra() are:

   poll   - successfully attach to remote data source and then exit
   rmon   - modify data to support unidiretional RMON stat reporting
   saslmech="mech" - specify a mandatory SASL mech
   TZ="tzset"   - specify a tzset(3) time zone specification
   xml   - print output in xml format.
   label="regex"   - match flow label with regex(3) regular expression.
   dsrs=dsrlist   - process these dsrs
      Where a dsrlist has the format:

[+/-]dsr[,[+/-]dsr]

Supported dsrs are:
    trans transport information, such as source id and seq number.
    flow   flow key data (proto, saddr, sport, dir, daddr, dport)
    time   time stamp fields (stime, ltime).
    metric basic ([s|d]bytes, [s|d]pkts, [s|d]rate, [s|d]load)
    agr   aggregation stats (trans, avgdur, mindur, maxdur, stdev).
    net   network objects (tcp, esp, rtp, icmp data).
    vlan   VLAN tag data
    mpls   MPLS label data
    jitter Jitter data ([s|d]jit, [s|d]intpkt)
    ipattr IP attributes ([s|d]ipid, [s|d]tos, [s|d]dsb, [s|d]ttl)
    psize packet size information
    mac   MAC addresses (smac, dmac)
    icmp   ICMP specific data (icmpmap, inode)
    encaps Flow encapsulation type indications
    tadj   Time adjustment data
    cor   Multi-probe correlation data
    cocode Country Codes
    asn   Autonomous System Number data
    suser src user captured data bytes (suser)
    duser dst captured user data bytes (duser)

Examples are:
  -M dsrs=time,flow,metric
  -M dsrs=-suser,-duser

   sql="select"   - use "select" as select clause in mysql call.

Illegal modes are not detectable by the standard library, and so unexpected results in command line parsing may occur if care is not taken with use of this option.

−n   Modify number to name converstion. This flag supports 4 states, specified by the modulus of the number of -n flags set. By default ra* programs do not provide hostname lookups, but they do lookup port and protocol names. The first **-n** will suppress port number to service conversion, **-nn** will suppress translation of protocol numbers to names (no lookups). **-nnn** will return you to full conversion, translating hostnames, port and protocol names, and **-nnnn** will return you to the default behavior. Because this indicator can be set in the .rarc file, multiple **-n** flags progress through the cycle.

−N <num>
    −N <start-end> −N <start+num> Process the first <num> records, the inclusive range <start - end > of records, or process <num + 1> records starting at start. These are the range of records that match the input filter if any filter is used.

−p <digits>
    Print <digits> number of units of precision for floating point values.

−q   Run in quiet mode. Configure Ra to not print out the contents of records. This can be used for a number of maintenance tasks, where you would be interested in the outcome of a program, or its progress, say with the -D option, without printing each input record.

−r [- | <file[::soffset:eoffset] ...>]
    Read data from <files> in the order presented on the commandline. '−' denotes stdin. If you want to read a set of files and then, when done, read stdin, use multiple occurences of the -r option. Ra can read **gzip(1)**, **bzip2(1)** and **compress(1)** compressed data files. Use the optional byte offset specification for reading specific data from a file.

−R <dir dir ...>
> Recursively decend the directory and process all the regular files that are encountered. The function does not decend to links, or directories that begin with '.'. The feature, like the -r command, does not do any file type checking.

−s <[-][[+[#]]field[:len] ...>
> Specify the **fields** to print. Ra uses a default printing field list, by specifying a field you can replace this list completely, or you can modify the existing default print list, using the optional '-' and '+[#]' form of the command. The available fields to print are:

> srcid, stime, ltime, sstime, dstime, dstime, dltime,
> trans, seq, flgs, dur, avgdur, stddev, mindur, maxdur,
> saddr, daddr, proto, sport, dport, stos, dtos, sdsb, ddsb,
> sco, dco, sttl, dttl, sipid, dipid, smpls, dmpls, svlan, dvlan,
> svid, dvid, svpri, dvpri, [s|d]pkts, [s|d]bytes,
> [s||d]appbytes, [s|d]load, [s|d]loss, [s|d]ploss, [s|d]rate,
> smac, dmac, dir, [s|d]intpkt, [s|d]intpktact, [s|d]intpktidl,
> [s|d]jit, [s|d]jitact, [s|d]jitidl, state, suser, duser,
> swin, dwin, trans, srng, erng, stcpb, dtcpb, tcprtt, inode,
> offset, spktsz, dpktsz, smaxsz, dmaxsz, sminsz, dminsz

srcid	argus source identifier.
stime	record start time
ltime	record last time.
trans	aggregation record count.
flgs	flow state flags seen in transaction.
seq	argus sequence number.
dur	record total duration.
avgdur	average duration of aggregated records..
stddev	standard deviation of aggregated duration times.
mindur	minimum duration of aggregated records.
maxdur	maximum duration of aggregated records.
smac	source MAC addr.
dmac	destination MAC addr.
saddr	source IP addr.
daddr	destination IP addr.
proto	transaction protocol.
sport	source port number.
dport	destination port number.
stos	source TOS byte value.
dtos	destination TOS byte value.
sdsb	source diff serve byte value.
ddsb	destination diff serve byte value.
sco	source IP address country code.
dco	destination IP address country code.
sttl	src -> dst TTL value.
dttl	dst -> src TTL value.
sipid	source IP identifier.
dipid	destination IP identifier.
smpls	source MPLS identifier.
dmpls	destination MPLS identifier.
pkts	total transaction packet count.
spkts	src -> dst packet count.

dpkts   dst -> src packet count.
bytes   total transaction bytes.
sbytes   src -> dst transaction bytes.
dbytes   dst -> src transaction bytes.
appbytes   total application bytes.
sappbytes   src -> dst application bytes.
dappbytes   dst -> src application bytes.
load   bits per second.
sload   source bits per second.
dload   destination bits per second.
loss   pkts retransmitted or dropped.
sloss   source pkts retransmitted or dropped.
dloss   destination pkts retransmitted or dropped.
ploss   percent pkts retransmitted or dropped.
psloss   percent source pkts retransmitted or dropped.
pdloss   percent destination pkts retransmitted or dropped.
rate   pkts per second.
srate   source pkts per second.
drate   destination pkts per second.
dir   direction of transaction
sintpkt (mSecs)
                   source interpacket arrival time
sintdist   source interpacket arrival time distribution
sintpktact (mSecs)
                   source active interpacket arrival time
sintdistact   source active interpacket arrival time
sintpktidl (mSecs)
                   source idle interpacket arrival time
sintdistidl   source idle interpacket arrival time
dintpkt (mSecs)
                   destination interpacket arrival time
dintdist   destination interpacket arrival time distribution
dintpktact (mSecs)
                   destination active interpacket arrival time
dintdistact   destination active interpacket arrival time distribution
dintpktidl (mSecs)
                   destination idle interpacket arrival time
dintdistidl   destination idle interpacket arrival time distribution
sjit   source jitter (mSecs).
sjitact   source active jitter (mSecs).
sjitidle   source idle jitter (mSecs).
djit   destination jitter (mSecs).
djitact   destination active jitter (mSecs).
djitidle   destination idle jitter (mSecs).
status   transaction status
suser   source user date buffer.
dvlan   destination user date buffer.
swin   source TCP window advertisement.
dwin   destination TCP window advertisement.
svlan   source VLAN identifier.
dvlan   destination VLAN identifier.
svid   source VLAN identifier.
dvid   destination VLAN identifier.

svpri	source VLAN priority.
dvpri	destination VLAN priority.
srng	start time for the filter timerange.
erng	end time for the filter timerange.
stcpb	source TCP base sequence number
dtcpb	destination TCP base sequence number
tcprtt	TCP connection setup round-trip time.
inode	ICMP intermediate node.
offset	record byte offset in file or stream.
spktsz	histogram for the src packet size distribution
smaxsz	maximum packet size for traffic transmitted by the src.
dpktsz	histogram for the dst packet size distribution
dmaxsz	maximum packet size for traffic transmitted by the dst.
sminsz	minimum packet size for traffic transmitted by the src.
dminsz	minimum packet size for traffic transmitted by the dst.

Examles are:
```
-s saddr     print only the source address.
-s -bytes    removes the bytes field from list.
-s +2srcid   adds the source identifier as the 2nd field.
-s spkts:18  prints src pkt count with a column width of 18.
-s smpls     print the local mpls label in the flow.
```

−S <host[:portnum]>

−S <URI://host[:portnum]> Specify a remote source of argus data. Read argus data using TCP or UDP sockets, using the URI format to indicate the type of argus record of interest (argus-tcp, argus-udp) and the source, as a name or an addresss. Use the optional ':portnum' to specify a port number other than the default; 561.

Examles are:
```
-S localhost             request via TCP argus records from localhost.
-S 192.168.0.4:12345     request via TCP argus records from 192.168.0.4, port 12345.
-S argus://anubis        request via TCP argus records from anubis, port 561.
-S argus-tcp://thoth:12345   request via TCP argus records from thoth, port 12345.
-S argus-udp://set:12345     request via UDP argus records from set, port 12345.
```

−t <timerange>

Specify the **<time range>** for matching **argus(5)** records. This option supports a high degree of flexibility in specifing explicit and relative time ranges with support for time field wildcarding.

The syntax for the **<time range>** is:
```
[timeComparisonInd]timeSpecification[-timeSpecification]
  timeComparisonInd: [x]i | n | c   (default = i)
    x negation   reverses the result of the time comparison
    i intersects match records that were active during this time period
    n includes   match records that start before and end after the period
    c contained  match records that start and end during the period

  timeSpecification: [[[yyyy/]mm/]dd.]HH[:MM[:SS]]
            [yyyy/]mm/dd
            %d{ymdHMS}
            seconds
            { + | - }%d{ymdHMS}
```

where '*' can be used as a wildcard.

Examples are:
  -t 14   specify the time range 2pm-3pm for today
  -t **.14   specify 2pm-3pm, every day this month
  -t 1270616652+2s all records that span 10/04/07.01:04:12 EDT.
  -t 1999y1m23d10h matches 10-11am on Jan, 23, 1999
  -t 10d*h*m15s   matches records that intersect the 15 sec,
                any minute, any hour, on the 10th of this month
  -t ****/11/23   all records in Nov 23rd, any year
  -t 23.11:10-14   11:10:00 - 2pm on the 23rd of this month
  -t -10m   matches 10 minutes before, to the present
  -t -1M+1d   matches the first day of the this month.
  -t -2h5m+5m   matches records that start before and end
               after the range starting 2 hours 5 minutes
               prior to the present, and lasting 5 minutes.

Time is compared using basic intersection operations. A record **iPntersects a specified time range if there is any intersection between the time range of the record and the comparison time range. This is the default behavior. A record includes the comparison time range if the intersection of the two ranges equals the comparison time, and a record is contained when the intersection equals the duration of the record. The comparison indicator is the first character of the range specification, without spaces.**

Examples are:
  -t n14:10:15-14:10:19 records include these 4s.
  -t c14:10-14:10:10   record starts and ends within these 10s.
  -t xi-5s+25s   record starts or ends 5 seconds earlier and
               20 seconds after 'now'.

−T \<secs\>
    Read **argus(5)** from remote server for **\<secs\>** of time.

−u Print time values using Unix time format (seconds from the Epoch).

−w \<file\> [filter-expression]
    Append matching data to **\<file\>**, in **argus** file format. An *output-file* of '-' directs **ra** to write the **argus(5)** records to stdout, allowing for "chaining" **ra*** style commands together. The optional filter-expression can be used to select specific output.

−X Don't read the default rarc file.

−z Modify status field to represent TCP state changes. The values of the status field when this is enabled are:
  's' - Syn Transmitted
  'S' - Syn Acknowledged
  'E' - TCP Established
  'f' - Fin Transmitted (FIN Wait State 1)
  'F' - Fin Acknowledged (FIN Wait State 2)
  'R' - TCP Reset

−Z <s|d|b>

>Modify status field to reprsent actual TCP flag values. <'s'rc | 'd'st | 'b'oth>. The characters that can be present in the status field when this is enabled are:

>'F' - Fin
>'S' - Syn
>'R' - Reset
>'P' - Push
>'A' - Ack
>'U' - Urgent Pointer
>'7' - Undefined 7th bit set
>'8' - Undefined 8th bit set

FILTER EXPRESSION

>If arguments remain after option processing, the collection is interpreted as a single filter **expression**. In order to indicate the end of arguments, a '−' is recommended before the filter expression is added to the command line.

>The filter expression specifies which **argus(5)** records will be selected for processing. If no *expression* is given, all records are selected, otherwise, only those records for which *expression* is 'true' will be printed.

>The syntax is very similar to the expression syntax for **tcpdump(1)**, as the tcpdump compiler was the basis for the **argus(5)** filter expression compiler. The semantics for **tcpdump(1)'s** packet filter expression are different when applied to transaction record filtering, so there are some major differences.

>The *expression* consists of one or more *primitives.* Primitives usually consist of an *id* (name or number) preceded by one or more qualifiers. There are three different kinds of qualifier:

>*type*   qualifiers say what kind of thing the id name or number refers to. Possible types are **srcid, encaps, host, net, port, tos, ttl, ptks, bytes, appbytes, data, rate, load, loss, ploss, mid, vid, vpri,** and **mid.**

>>E.g., 'srcid isis', 'encaps gre', 'host sphynx', 'net 192.168.0.0/16', 'port domain', 'ttl 1', 'ptks gt 2', 'ploss lt 5'. If there is no type qualifier, **host** is assumed.

>*dir*   qualifiers specify a particular transfer direction to and/or from *an id.* Possible directions are **src**, **dst**, **src or dst** and **src and dst**. E.g., 'src sphynx', 'dst net 192.168.0.0/24', 'src or dst port ftp', 'src and dst tos 0x0a', 'src or dst vid 0x12', 'dst vpri 0x02' . If there is no dir qualifier, **src or dst** is assumed.

>*proto*   qualifiers restrict the match to a particular protocol. Possible values are those specified in the **/etc/protocols** system file and a small number of extensions, (that should be defined but aren't). Specific extended values are **'ipv4'**, (to specify just ip version 4), in contrast to the defined proto **'ipv6'**. The defined proto **'ip'** reduces to the filter 'ipv4 or ipv6'.

>>When preceded by *ether*, the protocol names and numbers that are valid are specified in ./include/ethernames.h.

>In addition to the above, there are some special 'primitive' keywords that don't follow the pattern: **gateway**, **multicast**, and **broadcast**. All of these are described below.

>More complex filter expressions are built up by using the words **and**, **or** and **not** to combine primitives. E.g., 'host foo and not port ftp and not port ftp-data'. To save typing, identical qualifier lists can be

omitted. E.g., 'tcp dst port ftp or ftp-data or domain' is exactly the same as 'tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain'.

Allowable primitives are:

srcid *argusid*
> True if the argus identifier field in the Argus record is *srcid*, which may be an IP address, a name or a decimal/hexidecimal number.

encaps *type*
> True if the encapsulation used by the flow in the Argus record includes the *type*. The list of valid encapsulation types is:

> mpls, eth, 802q, llc, pppoe, isl, gre, ah, ipnip, ipnip6, chdlc

dst host *host*
> True if the IP destination field in the Argus record is *host*,
> which may be either an address or a name.

src host *host*
> True if the IP source field in the Argus record is *host*.

host *host*
> True if either the IP source or destination in the Argus record is *host*.
> Any of the above host expressions can be prepended with the keywords
> **ip**, **arp**, or **rarp** as in:
>> **ip** host *host*
> which is equivalent to:
>> **ether proto ip and host** *host*
> If *host* is a name with multiple IP addresses, each address will be checked for a match.

ether dst *ehost*
> True if the ethernet destination address is *ehost*. *Ehost* may be either a name from /etc/ethers or a number (see *ethers*(3N) for numeric format).

ether src *ehost*
> True if the ethernet source address is *ehost*.

ether host *ehost*
> True if either the ethernet source or destination address is *ehost*.

gateway *host*
> True if the transaction used *host* as a gateway. I.e., the ethernet source or destination address was *host* but neither the IP source nor the IP destination was *host*. *Host* must be a name and must be found in both /etc/hosts and /etc/ethers. (An equivalent expression is
>> **ether host** *ehost* **and not host** *host*
> which can be used with either names or numbers for *host / ehost*.)

dst net *cidr*
> True if the IP destination address in the Argus record matches the *cidr* address.

src net *cidr*
> True if the IP source address in the Argus record matches the *cidr* address.

net *cidr*

>   True if either the IP source or destination address in the Argus record matches *cidr* address.

dst port *port*

>   True if the network transaction is ip/tcp or ip/udp and has a destination port value of *port*. The
>   *port* can be a number or a name used in /etc/services (see *tcp*(4P) and *udp*(4P)). If a name is used,
>   both the port number and protocol are checked. If a number or ambiguous name is used, only the
>   port number is checked (e.g., **dst port 513** will print both tcp/login traffic and udp/who traffic, and
>   **port domain** will print both tcp/domain and udp/domain traffic).

src port *port*

>   True if the network transaction has a source port value of *port*.

port *port*

>   True if either the source or destination port in the Argus record is *port*. Any of the above port
>   expressions can be prepended with the keywords, **tcp** or **udp**, as in:
>
>   >   **tcp src port** *port*
>
>   which matches only tcp connections.

ip proto *protocol*

>   True if the Argus record is an ip transaction (see *ip*(4P)) of protocol type *protocol*. *Protocol* can
>   be a number or any of the string values found in */etc/protocols*.

multicast

>   True if the network transaction involved an ip multicast address. By specifing ether multicast, you
>   can select argus records that involve an ethernet multicast address.

broadcast

>   True if the network transaction involved an ip broadcast address. By specifing ether broadcast,
>   you can select argus records that involve an ethernet broadcast address.

ether proto *protocol*

>   True if the Argus record is of ether type *protocol*. *Protocol* can be a number or a name like *ip*,
>   *arp*, or *rarp*.

[src | dst] ttl [gt | gte | lt | lte | eq] *number*

>   True if the TTL in the Argus record equals *number*.

[src | dst] tos [gt | gte | lt | lte | eq] *number*

>   True if the TOS in the Argus record (default) equals *number*.

[src | dst] vid [gt | gte | lt | lte | eq] *number*

>   True if th VLAN id in the Argus record (default) equals *number*.

[src | dst] vpri [gt | gte | lt | lte | eq] *number*

>   True if the VLAN priority in the Argus record (default) equals *number*.

[src | dst] mid [gt | gte | lt | lte | eq] *number*

>   True if the MPLS Label in the Argus record (default) equals *number*.

[src | dst] pkts [gt | gte | lt | lte | eq] *number*

>   True if the packet count in the Argus record (default) equals *number*.

[src | dst] bytes [gt | gte | lt | lte | eq] *number*
> True if the byte count in the Argus record (default) equals *number*.

[src | dst] appbytes [gt | gte | lt | lte | eq] *number*
> True if the application byte count in the Argus record (default) equals *number*.

[src | dst] rate [gt | gte | lt | lte | eq] *number*
> True if the rate in the Argus record (default) equals *number*.

[src | dst] load [gt | gte | lt | lte | eq] *number*
> True if the load in the Argus record (default) equals *number*.

Ra filter expressions support primitives that are specific to flow states and can be used to select flow records that were in these states at the time they were generated. *normal*, *wait*, *timeout*, *est* or *con*

Primitives that select flows that experienced fragmentation. *frag* and *fragonly*

Support for selecting flows that used multiple pairs of MAC addresses during their lifetime. *multipath*

Primitives specific to TCP flows are supported. *syn*, *synack*, *ecn*, *fin*, *finack*, *reset*, *retrans*, *outoforder* and *winshut*

Primitives specific to ICMP flows are supported. *echo*, *unreach*, *redirect* and *timexed*

For some primitives, a direction qualifier is appropriate. These are *frag*, *reset*, *retrans*, *outoforder* and *winshut*

Primitives may be combined using:

> A parenthesized group of primitives and operators (parentheses are special to the Shell and must be escaped).

> Negation ('**!**' or '**not**').

> Concatenation ('**and**').

> Alternation ('**or**').

Negation has highest precedence. Alternation and concatenation have equal precedence and associate left to right. Note that explicit **and** tokens, not juxtaposition, are now required for concatenation.

If an identifier is given without a keyword, the most recent keyword is assumed. For example,
> **not host sphynx and anubis**

is short for
> **not host sphynx and host anubis**

which should not be confused with
> **not (host sphynx or anubis)**

Expression arguments can be passed to **ra(1)** as either a single argument or as multiple arguments, whichever is more convenient. Generally, if the expression contains Shell metacharacters, it is easier to pass it as

a single, quoted argument. Multiple arguments are concatenated with spaces before being parsed.

Startup Processing

Ra begins by searching for the configuration file **.rarc** first in the directory, **$ARGUSHOME** and then **$HOME**. If a **.rarc** is found, all variables specified in the file are set.

Ra then parses its command line options and set its internal variables accordingly.

If a configuration file is specified on the command-line, using the "-f <confile>" option, the values in this .rarc formatted file superceed all other values.

EXAMPLES

To report all TCP transactions from and to host 'narly.wave.com', reading transaction data from *argus-file* argus.data:

ra -r argus.data - tcp and host narly.wave.com

Create the *argus-file* icmp.log with all ICMP events involving the host nimrod, using data from *argus-file*, but reading the transaction data from *stdin*:

cat *argus-file* **| ra -r - -w icmp.log - icmp and host nimrod**

OUTPUT FORMAT

The following is a brief description of the default output of .B ra. While this is by no means the 'preferred' set of data that one should generate, it represents a starting point for using flow data in general. This also looks pretty good on 80 column terminals. The format is:

time proto srchost dir dsthost metrics state

time

The format of the *time* field is specified by the .rarc file, using syntax supported by the routine **strftime(3V).** The default is '%T'. **Argus** transactional data contains both starting and ending transaction times, with precision to the microsecond. However, **ra** by default prints out the 'stime' field, the records starting time.

proto [options protocol]

The *proto* indicator consists of two fields. The first is protocol specific and the designations are:

```
T     - Time Corrected/Adjusted
*     - Multiple sub-IP encapsulations
e     - Ethernet encapsulated flow
M     - Multiple mac addresses seen
m     - MPLS encapsulated flow
l     - LLC encapsulated flow
v     - 802.11Q encapsulations/tags
w     - 802.11 wireless encapsulation
p     - PPP over Enternet encapsulated flow
i     - ISL encapsulated flow
G     - GRE encapsulation
a     - AH encapsulation
P     - IP tunnel encapsulation
6     - IPv6 tunnel encapsulation
H     - HDLC encapsulation
C     - Cisco HDLC encapsulation
```

```
        A      -  ATM encapsulation
        S      -  SLL encapsulation
        F      -  FDDI encapsulation
        s      -  SLIP encapsulation
        R      -  ARCNET encapsulation
        I    -  ICMP events mapped to this flow
        U      -  ICMP Unreachable event mapped to this flow
        R      -  ICMP Redirect event mapped to this flow
        T      -  ICMP Time Exceeded mapped to this flow
        *    -  Both Src and Dst loss/retransmission
        s    -  Src loss/retransmissions
        d    -  Dst loss/retransmissions
        &    -  Both Src and Dst packet out of order
        i    -  Src packets out of order
        r    -  Dst packets out of order
        @    -  Both Src and Dst Window Closure
        S    -  Src TCP Window Closure
        D    -  Dst TCP Window Closure
        *    -  Silence suppression used by both src and dst (RTP)
        s    -  Silence suppression used by src
        d    -  Silence suppression used by src
        E   -  Both Src and Dst ECN
        x   -  Src Explicit Congestion Notification
        t   -  Dst ECN
        V  -  Fragment overlap seen (if fragments seen)
        f  -  Partial Fragment (if fragments seen)
        F  -  Fragments seen
        O  -  multiple IP options set
        S  -  IP option Strict Source Route
        L  -  IP option Loose Source Route
        T  -  IP option Time Stamp
        +  -  IP option Security
        R  -  IP option Record Route
        A  -  IP option Router Alert
        U  -  unknown IP options set
```

The second field indicates the upper protocol used in the transaction. This field will contain the first 4 characters of the official name for the protocol used, as defined in RFC-1700. Argus attempts to discovery the Realtime Transport Protocol, when it is being used. When it encounters RTP, it will indicate its use in this field, with the string 'rtp'. Use of the **-n** option, twice (-nn), will cause the actual protocol number to be displayed.

srchost

The *srchost* field is meant to convey the originator of the data in the flow. This field is protocol dependent, and for IP protocols will contain the src IP address/name. For TCP and UDP, the field will also contain the port number/name, separated by a period.

The 'src' is generally the entity that first transmits a packet that is a part of a flow. However, the assignment of 'src' and 'dst' semantics is somewhat complicated by the notion of loss, or half-duplex monitoring, especially when connection-oriented protocol , such as TCP, are reported. In this case the 'src' is the entity that initiated the flow.

dir

The *dir* field will have the direction of the transaction, as can be best determined from the datum, and is used to indicate which hosts are transmitting. For TCP, the dir field indicates the actual source of the TCP connection, and the center character indicating the state of the transaction.

- - transaction was NORMAL
| - transaction was RESET
o - transaction TIMED OUT.
? - direction of transaction is unknown.

dsthost

The *dsthost* field is meant to convey the recipient of the data in the flow. Like the srchost field, this field is protocol dependent, and for IP protocols will contain the dst IP address/name, and optionally the DSAP.

metrics

metrics represent the general sets of fields that reflect the activity of the flow. In the default output, there are 4 fields. The first 2 are the packet counts and the last 2 are the byte counts for the specific transaction. The fields are paired with the previous host fields, and represent the packets transmitted by the respective host.

state

The *state* field indicates the principle state for the transaction report, and is protocol dependent. For all the protocols, except ICMP, this field reports on the basic state of a transaction.

REQ|INT (requested|initial)

This indicates that this is the *initial* state report for a transaction and is seen only when the *argus-server* is in DETAIL mode. For TCP connections this is **REQ**, indicating that a connection is being requested. For the connectionless protocols, such as UDP, this is **INT**.

ACC (accepted)

This indicates that a request/response condition has occurred, and that a transaction has been detected between two hosts. For TCP, this indicates that a connection request has been answered, and the connection will be accepted. This is only seen when the *argus-server* is in DETAIL mode. For the connectionless protocols, this state indicates that there has been a single packet exchange between two hosts, and could qualify as a request/response transaction.

EST|CON (established|connected)

This record type indicates that the reported transaction is active, and has been established or is continuing. This should be interpreted as a state report of a currently active transaction. For TCP, the EST state is only seen in DETAIL mode, and indicates that the three way handshake has been completed for a connection.

CLO (closed)

TCP specific, this record type indicates that the TCP connection has closed normally.

TIM (timeout)

Activity was not seen relating to this transaction, during the **argus** server's timeout period for this protocol. This state is seen only when there were packets recorded since the last report for this transaction.

For the ICMP and ICMPv6 protocols, the *state* field displays specific aspects of the ICMP type. ICMP state can have the values:

ECO	Echo Request
ECR	Echo Reply
SRC	Source Quench
RED	Redirect
RTA	Router Advertisement
RTS	Router Solicitation
TXD	Time Exceeded
PAR	Parameter Problem
TST	Time Stamp Request
TSR	Time Stamp Reply
IRQ	Information Request
IRR	Information Reply
MAS	Mask Request
MSR	Mask Reply
URN	Unreachable network
URH	Unreachable host
URP	Unreachable port
URF	Unreachable need fragmentation
URS	Unreachable source failed
URNU	Unreachable dst network unknown
URHU	Unreachable dst host unknown
URISO	Unreachable source host isolated
URNPRO	Unreachable network administrative prohibited
URHPRO	Unreachable host administrative prohibited
URNTOS	Unreachable network TOS prohibited
URHTOS	Unreachable host TOS prohibited
URFIL	Unreachable administrative filter
URPRE	Unreachable precedence violation
URCUT	Unreachable precedence cutoff
MRQ	Membership Query
MHR	Membership Report
NRS	Neighbor Discovery Router Solicit
NRA	Neighbor Discovery Router Advertisement
NNS	Neighbor Discovery Neighbor Solicit
NNA	Neighbor Discovery Neighbor Advertisement
PTB	Packet Too Big

OUTPUT EXAMPLES

These examples show typical **ra** output, and demonstrates a number of variations seen in **argus** data. This **ra** output was generated using the **-n** option to suppress number translation.

Thu 12/29 06:40:32 S tcp 132.3.31.15.6439 -> 12.23.14.77.23   CLO

This is a normal tcp transaction to the telnet port on host 12.23.14.77. The IP Option strict source route was seen.

Thu 12/29 06:40:32   tcp 132.3.31.15.6200 <| 12.23.14.77.25   RST

This tcp transaction from the smtp port of host 12.23.14.77 was **RESET**. In many cases this indicates that the transaction was rejected, however some os's will use RST to close an active TCP. Use either the -z or -Zb options to specify exactly what conditions existed during the connection.

Thu 12/29 03:39:05 M igmp 12.88.14.10   <-> 128.2.2.10   CON

This is an igmp transaction state report, usually seen with MBONE traffic. There was more than one source and destination MAC address pair used to support the transaction, suggesting a possible routing loop.

Thu 12/29 06:40:05 * tcp 12.23.14.23.1043 <-> 12.23.14.27.6000 TIM

This is an X-windows transaction, that has **TIMEDOUT**. Packets were retransmitted during the connection.

Thu 12/29 07:42:09   udp 12.9.1.115.2262 -> 28.12.141.6.139 INT

This is an initial netbios UDP transaction state report, indicating that this is the first datagram encountered for this transaction.

Thu 12/29 06:42:09   icmp 12.9.1.115   <-> 12.68.5.127   ECO

This example represents a "ping" of host 12.9.1.115, and its response.

This next example shows the **ra** output of a complete TCP transaction, with the preceeding Arp and DNS requests, while reading from a remote *argus-server*. The '*' in the CLO report indicates that at least one TCP packet was retransmitted during the transaction. The hostnames in this example are ficticious.

```
% ra -S argus-tcp://argus-server and host i.qosient.com
ra: Trying argus-server port 561
ra: connected Argus Version 3.0
Sat 12/03 15:29:38    arp i.qosient.com    who-has dsn.qosient.com  INT
Sat 12/03 15:29:39    udp i.qosient.com.1542 <->   dns.qosient.53  INT
Sat 12/03 15:29:39    arp i.qosient.com    who-has qosient.com      INT
Sat 12/03 15:29:39 *  tcp i.qosient.com.1543  ->   qosient.com.smtp CLO
```

AUTHORS

Carter Bullard (carter@qosient.com).

FILES

/etc/ra.conf

SEE ALSO

argus(8) **tcpdump**(1),

Postel, Jon, *Internet Protocol*, RFC 791, Network Information Center, SRI International, Menlo Park, Calif., May 1981.

Postel, Jon, *Internet Control Message Protocol*, RFC 792, Network Information Center, SRI International, Menlo Park, Calif., May 1981.

Postel, Jon, *Transmission Control Protocol*, RFC 793, Network Information Center, SRI International, Menlo Park, Calif., May 1981.

Postel, Jon, *User Datagram Protocol*, RFC 768, Network Information Center, SRI International, Menlo Park, Calif., May 1980.

McCanne, Steven, and Van Jacobson, *The BSD Packet Filter: A New Architecture for User-level Capture*, Lawrwnce Berkeley Laboratory, One Cyclotron Road, Berkeley, Calif., 94720, December 1992.